Assignment 2 - MAS6024

Student Registration Number: 190186373

Contents

Part 1

Given that we know the size of the sample n and that the managing director can then observe the amount of defective leaflets in that sample, we can define a function that will return the proportion of defective leaflets in the random sample. Our function is called `prop_defect` and will take function arguments n, p_1, p_2, N_1 and N_2.

Let us describe each line of the function definition. Using `rbinom`, we can model printing device i as a set of N_i realisations of a binomial distribution, namely `device1` and `device2`. Each realisation represents a leaflet. This means that any given realisation takes the value 1 (a success) with probability p_i and 0 (a failure) with probability $1 - p_i$. We can concatenate the sets for both devices, creating a set called `all_leaflets`, of length $N_1 + N_2$. This model now accounts for the fact that when we take a random sample, we will not know which leaflet came from a particular device.

```
prop_defect <- function(n = 200, N1 = 10000, N2 = 20000, p1 = 0.05, p2 = 0.02){
  device1 <- rbinom(N1, 1, p1)
  device2 <- rbinom(N2, 1, p2)
  all_leaflets <- c(device1, device2)
  ...
```

Now, we can define `nsample`, which is a random sample of `all_leaflets`, of size n. Using the `sum()` function on `nsample`, we can find the number of defective leaflets in the sample. Defective leaflets are represented by the value 1 in the vector, the only non-zero entries. Finally, we can calculate and return the proportion of defective leaflets in the sample by dividing our amount of defective leaflets by the sample size `n`. With this function, we return a probability estimate, say $\hat{p}$, of p based on our random sample.

```
  ...
  nsample <- sample(all_leaflets, size = n)
  sample_defected <- sum(nsample)
  prop <- sample_defected/n
  return(prop)
}
```

The full function is defined below.

```
prop_defect <- function(n = 200, N1 = 10000, N2 = 20000, p1 = 0.05, p2 = 0.02){
  device1 <- rbinom(N1, 1, p1)
  device2 <- rbinom(N2, 1, p2)
  all_leaflets <- c(device1, device2)
```

```r
  nsample <- sample(all_leaflets, size = n)
  sample_defected <- sum(nsample)
  prop <- sample_defected/n
  return(prop)
}
```

Part 2

To begin describing the probability distribution of the proportion of defective leaflets in a random sample, we can input the given values for n, p_1, p_2, N_1 and N_2 into our function.

```r
prop_defect(n = 200, N1 = 10000, N2 = 20000, p1 = 0.05, p2 = 0.02)
```

We can now use Monte Carlo simulation on our expression to analyse the range of probabilities estimates. It is advisable to use as many simulations as time permits whilst maximising accuracy. A simulation count of $m = 10000$ should ensure this.

```r
set.seed(373)
mrep <- 10000
prop_estimates <- replicate(mrep, expr = prop_defect(n = 200, p1 = 0.05, p2 = 0.02,
                                                     N1 = 10000, N2 = 20000))
cat('| Minimum:', min(prop_estimates), '| Maximum', max(prop_estimates), '| Mean:',
    mean(prop_estimates), '| Stan. Dev.', sd(prop_estimates), '\n')
```

```
## | Minimum: 0 | Maximum 0.08 | Mean: 0.029948 | Stan. Dev. 0.01208519
```

Using the quantile function, we can see that 95% of our estimates in the interval (0.01, 0.055). Also, from ‘prop_agree_to2dp, we can see that only 33% of our estimates agree to within 2 decimal places of the mean.

```r
set.seed(373)
quantile(prop_estimates, c(0.025, 0.975))
```

```
##  2.5% 97.5%
## 0.010 0.055
```

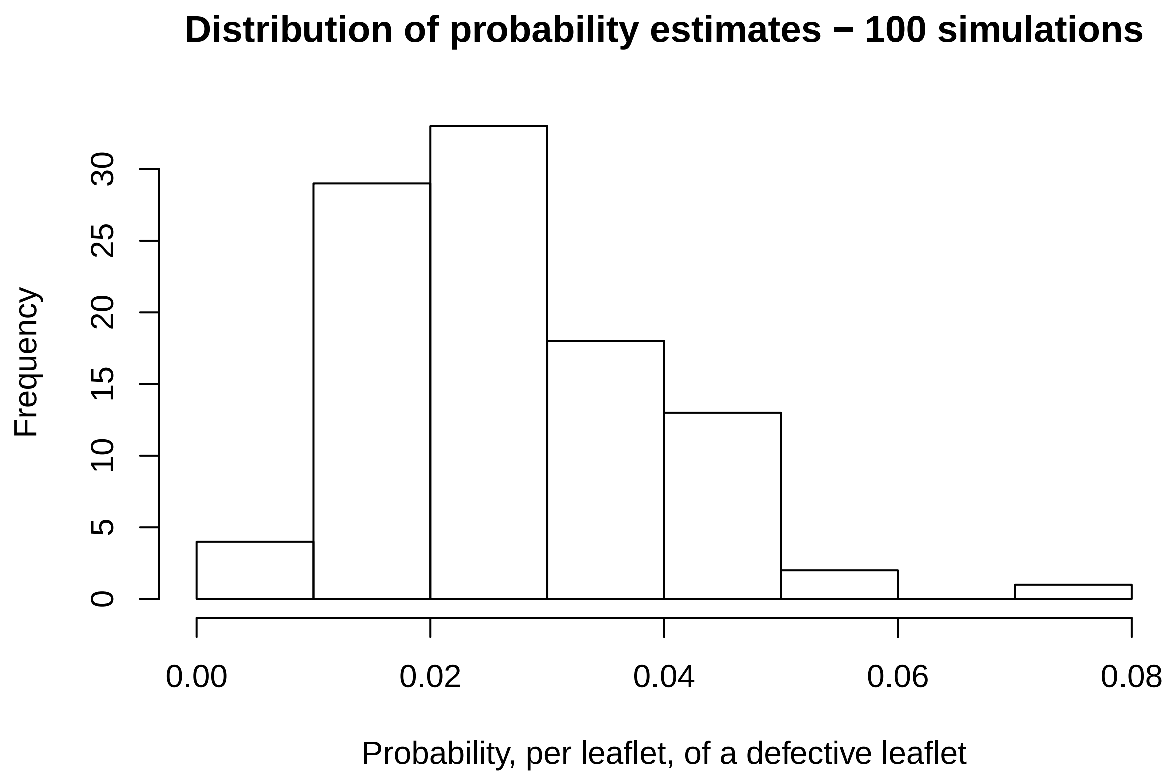
```r
prop_agree_to2dp <- sum(prop_estimates < 0.035 & prop_estimates >= 0.025 )/mrep
prop_agree_to2dp
```

```
## [1] 0.3263
```

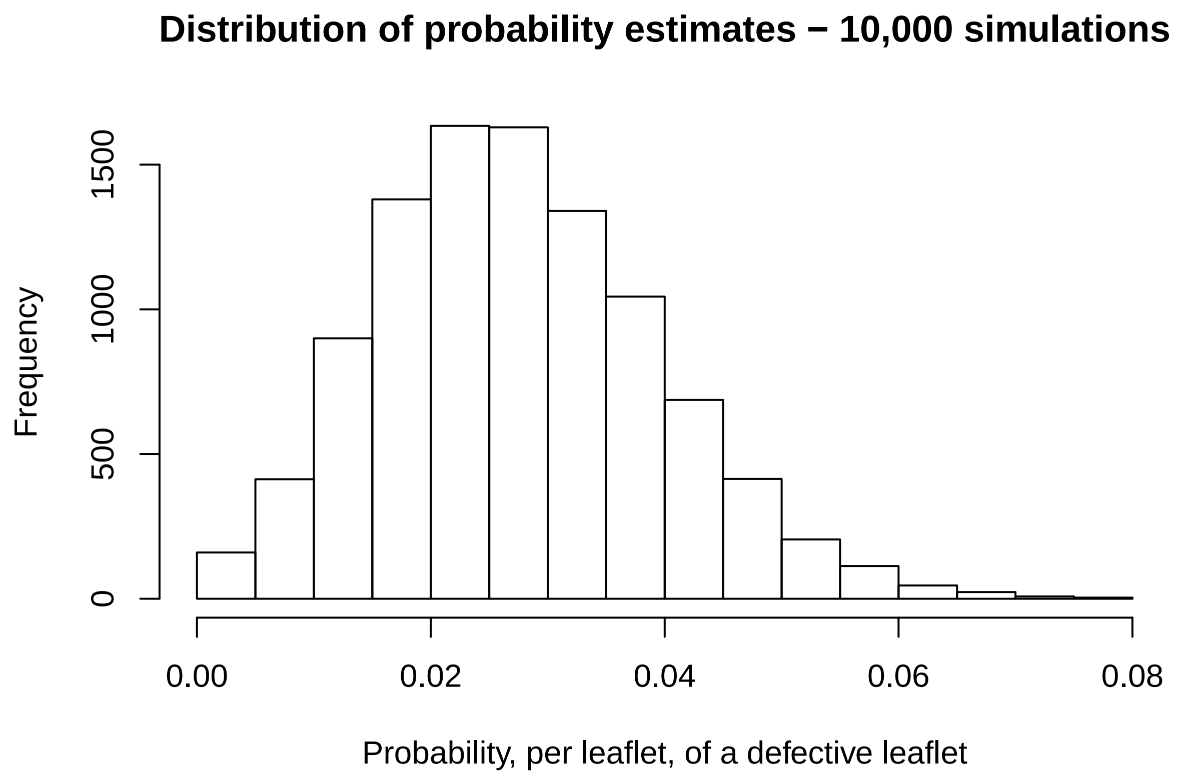
```r
prop_est2 <- replicate(100, expr = prop_defect(n = 200, p1 = 0.05, p2 = 0.02,
                                                     N1 = 10000, N2 = 20000))
```

We can see from the R output that our mean probability estimate, say $\hat{p}_{n=200}$, is 0.03 to 2 decimal places. This means we can say with reasonable accuracy that the probability of a leaflet being defective, in this case, is 0.03. Despite this, there is a lot of variability in the data that we need to consider. As detailed earlier, 95% of our values are in the interval (0.010, 0.055) and only 33% of our estimates agree to within 2 decimal places of the mean. This suggests we have a large range of probabilities, which we can visualise using a histogram. We can also demonstrate the need for a large amount of simulations with histograms.

```r
hist(prop_est2,
     main = 'Distribution of probability estimates - 100 simulations',
     xlab = 'Probability, per leaflet, of a defective leaflet')
```



```r
hist(prop_estimates,
     main = 'Distribution of probability estimates - 10,000 simulations',
     xlab = 'Probability, per leaflet, of a defective leaflet')
```



The first histogram does not give us a good indication of the true distribution of the probability. With 10,000 simulations, the second histogram indicates that the distribution is a skewed bell-shaped curve centered around 0.3. Even with 10,000 realisations we can see a wide bell-shape, suggesting high variability in the probability of producing a defective leaflet.

Part 3

Under our new assumption $p_1 = p_2 = p$, we can use our function `prop_defect` to find a new $\hat{p}$. We will eventually replicate this to help us calculate $Pr(|p - \hat{p}| < \epsilon) > 0.95$. To determine whether our $\hat{p}$ lies within ϵ of our true probability p, we can use a Boolean expression in R, equivalent to $|p - \hat{p}| < \epsilon$:

```
p_hat <- prop_defect(n, p1=p, p2=p)
(abs(p - p_hat) < epsilon)
```

Let us define a function `prob_within_epsilon`, with function arguments ϵ, p, N_1, N_2, and n. I have fixed the default values for the arguments to match the values described in question 4.

```
prob_within_epsilon <- function(epsilon = 0.05, p = 0.1, N1 = 10000, N2 = 20000, n){
  rep_within_epsilon <- replicate(5000,
                                  expr = (abs(p - prop_defect(n, p1=p, p2=p)) < epsilon))
  prob <- sum(rep_within_epsilon)/5000
  return(prob)
}
```

Using the `replicate` function, we can perform a Monte Carlo simulation on our Boolean expression. Let us call the set of these replications `rep_within_epsilon`. Note that I have used `prop_defect(n, p1=p, p2=p)` rather than more simply specifying it as $\hat{p}$. This is to ensure that when we replicate the full expression, it doesn't use the same pseudo-randomization of $\hat{p}$. A simulation count of 5000 has been used as it is the largest amount before the code begins to take too long to run. We can be reasonably confident of our predictions with a simulation count as high as 5000.

Now that we have simulated our Boolean, we can find the proportion of `TRUE` values and ask the function to return this proportion. This is our estimate of $Pr(|p-\hat{p}| < \epsilon) > 0.95$. If we use trial and error by running this function with different values of n, we can find the smallest n such that $Pr(|p-\hat{p}| < \epsilon) > 0.95$, as required.

Part 4

Using our function from part 3, we can test different n to find the smallest n such that $Pr(|p-\hat{p}| < \epsilon) > 0.95$. Let us try n = 50, 100, 125, 140.

```
set.seed(373)
prob_within_epsilon(n=50)
```

```
## [1] 0.762
```

```
prob_within_epsilon(n=100)
```

```
## [1] 0.9032
```

```
prob_within_epsilon(n=125)
```

```
## [1] 0.9306
```

```
prob_within_epsilon(n=140)
```

```
## [1] 0.9474
```

For $n = 125$, our probability is 0.9234. For $n = 140$, our probability is 0.9492. Let us try values in between.

```
set.seed(373)
prob_within_epsilon(n=135)
```

```
## [1] 0.9576
```

```
prob_within_epsilon(n=134)
```

```
## [1] 0.955
```

```
prob_within_epsilon(n=133)
```

```
## [1] 0.9374
```

For $n = 134$, our probability is 0.9516. For $n = 133$, our probability is 0.9472. Therefore, for our simulations, the smallest n in such that $Pr(|p-\hat{p}| < \epsilon) > 0.95$ is $n = 134$. It is important to note that this value for n can vary depending on the pseudo-randomization. This is a natural side-effect of Monte Carlo simulation. As I have set a random seed, my results should be reproducible.

Part 5

There are two restrictions on p and ϵ that are important for answering this question:

- $\epsilon \geq 0$ (negative values for ϵ yield trivial solutions for this question)
- $0 \leq p < 1$ (by simple probability law)

Firstly, if we want to find values of p and ϵ such that $Pr(|p-\hat{p}| < \epsilon) = 1$, we need to think about when it is impossible for $\hat{p}$ to not to be within ϵ of p. Since $0 \leq p < 1$, we can be sure that $0 \leq |p-\hat{p}| < 1$. This means that for any $\epsilon \geq 1$ other than $\epsilon = 0$, we have $Pr(|p-\hat{p}| < \epsilon) = 1$.

Now, let us consider values of p that will satisfy $Pr(|p - \hat{p}| < \epsilon) = 1$. When $\epsilon > 0$ and $p = \hat{p}$, our condition will clearly be satisfied. It is only possible that $p = \hat{p}$ when $p = 0$ or $p = 1$, where every leaflet will be defective ($p = 1$) or no leaflet will be defective ($p = 0$). In summary, $Pr(|p - \hat{p}| < \epsilon) = 1$ is satisfied when $p = 0$, $p = 1$ or $\epsilon \geq 1$.

If we are interested in values of n that cannot satisfy $Pr(|p - \hat{p}| < \epsilon) > 0$, we are effectively looking for values of n such that $Pr(|p - \hat{p}| < \epsilon) = 0$. If this is true, then $|p - \hat{p}| \geq \epsilon$. As the sample size increases, it is likely that $|p - \hat{p}|$ will decrease, since there will be more accuracy in our prediction. This means that there are more values of ϵ that will satisfy our condition.

In the extreme case n=1, $\hat{p}$ is either 0 or 1, so for a reasonable ϵ, it is unlikely that that $\hat{p}$ will be within ϵ. Despite this, there are no values of n that will guarantee $Pr(|p - \hat{p}| < \epsilon) > 0$ for all p and ϵ.